PC ADDA-14 CARD

FPC-011

USER'S MANUAL

TABLE OF CONTEST

1. INTRODUCTION 1
2. SPECIFICATION 1
3. PACKING 1
4. OPERATING PROCEDURE 1
5. D-TYPE CONNECTOR PINOUT 3
6. FLOW CHART OF THE ADDA-14 (BASIC) 4
7. BASIC PROGRAM LIST 5
8. INTRODUCTION TO AD14 ASSEMBLY LANGUAGE 6
9. FLOW CHART OF AD14 (ASSEMBLY) 7
10. ASSEMBLY LANGUAGE PROGRAM LIST 8
11. THE THERMAL SENSOR 12
12. PROCEDURE TO ADJUST THE WORKING RANGE 13

1. INTRODUCTION:

14 bit ADDA CARD (PC PC/XT PC/AT) is a high precision data conversion system. It contain 2 channel 14 bit Digital to analog (Setting SW1, SW2 for select unipolar or bipolar) and 16 channel 14 bit analog to digital conversion (unipolar or biploar)

2. SPECIFICATION:

A/D

* 14 bits resolution
* 16 input channels
* Unipolar (0-8.5V) or Bipolar (-8.5V - +8.5V) input level
* Conversion time less than 42 usec
* Relative Accuracy (25°C) $\pm\frac{1}{2}$ LSB max
* Temperature Coefficients 88 ppm/°C

D/A

* 14 bits resolution
* 2 channels output (one standard, one option)
* Unipolar (0-8.5V) or Bipolar (-8.5V - +8.5V) output
* Current setting time less 0.5 usec

3. PACKING :

* ADDA 14 adapter (4 layer board)
* Complete user's manual
* Demonstration software diskette

4. OPERATING PROCEDURE:

1) Jumper Setting: (defaul SW4)

Position 1: select port address from $170-$17F (decimal (UP) 368 - 383)

Position 2: select port address from $1F0-$1FF (decimal (DOWN) 496 - 511)

2) Potential resistor : adjust the A/D and D/A full scale reference voltage, normal at 8.5V.

3) port = 368 (or 496)
   port + 0 = output data (0-15) to select A/D channel.
   port + 1 = output $00 to clear A/D register
   port + 2 = read A/D low 8 bit data (bit 0 - bit 7)
   port + 3 = read A/D high 6 bit data
   port + 4 = output DA0 low 8 bit data
   port + 5 = output DA0 high 6 bit data
   port + 6 = output DA1 low 8 bit data
   port + 7 = output DA1 high 6 bit data
   port + 8 = loop back 8 times to start AD high 7 bit conversion
   port + 12 = loop back 8 times to start AD low 7 bit conversion

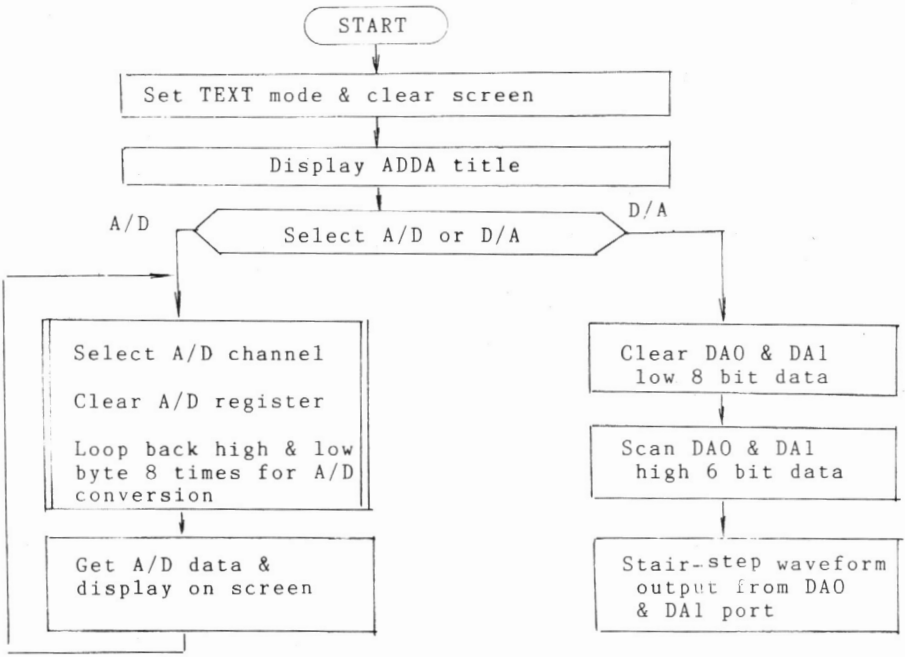
4) Analog to digital conversion procedure
   a. output channel number to port + 0
   b. Initial register using softswitch in port + 1 to clear previous data
   c. start convertion using softswitch in port +8, port+ 12 exactly 8 times individually
   d. read data in port + 2 for low 8 bit data and port + 3 for high 6 bit data

5) Digital to analog conversion :
   a. Output DA0 high 6 bit data to port + 5
   b. Output DA0 low 8 bit data to port + 4
   c. Output DA1 high 6 bit data to port + 7
   d. Output DA1 low 8 bit data to port + 6

5. D-TYPE CONNECTOR PINOUT

PIN	SINGAL	PIN	SIGNAL
1	+12V	14	-12V
2	DA1 out	15	DA0 out
3	GND	16	CH 15
4	CH 14	17	CH 13
5	CH 12	18	CH 11
6	CH 10	19	CH 9
7	CH 8	20	CH 7
8	CH 6	21	CH 5
9	CH 4	22	CH 3
10	CH 2	23	CH 1
11	CH 0	24	GND
12	GND	25	- 5V
13	+5V		

6. Flow chart of the ADDA-14 (BASIC LANGUAGE)



7. BASIC PROGRAM LIST

```
10 WIDTH 80: SCREEN 0,0,0: KEY OFF: CLS: PORT = &H170
20 LOCATE 5,18:PRINT "14 BIT AD-DA CONVERTION CARD"
30 LOCATE 6,18:PRINT "============================="
40 LOCATE 9,20:PRINT "1, D/A CONVERTION DEMO"
50 LOCATE 11,20:PRINT "2, A/D CONVERTION DEMO"
60 A$=INKEY$:IF A$="" THEN 60
70 IF A$="1" THEN 200
80 IF A$="2" THEN 400
90 GOTO 10
200 CLS
202 LOCATE 5,15:PRINT"D/A CONVERTION DEMO"
204 LOCATE 7,15:PRINT "OUTPUT WAVEFORM FROM D/A PORT"
206 LOCATE 9,15:PRINT "PRESS ANY KEY RETURN MENU"
210 OUT PORT +4,0: OUT PORT + 6,0
220 FOR I= 0 TO 63
230 OUT PORT+5,I: OUT PORT+7,I
240 NEXT I
250 A$=INKEY$:IF A$="" THEN 210
260 GOTO 10
400 CLS
410 FOR CHANNEL=0 TO 15
420 GOSUB 550
430 LB = INP ( PORT + 2)
440 HB = INP ( PORT + 3)
450 DA = ( HB - 64 * (INT ( HB / 64 ))) * 256 + LB
460 PRINT "   CHANNEL= ";CHANNEL,"DATE= ";DA
470 NEXT CHANNEL
480 PRINT:PRINT :PRINT
490 GOTO 410
550 OUT PORT+1,0
560 OUT PORT+0,CHANNEL
570 FOR I=1 TO 8:A=INP(PORT+8):NEXT I
580 FOR I=1 TO 8:A=INP(PORT+12):NEXT I
590 RETURN
```

8. INTRODUCTION TO AD14 ASSEMBLY LANGUAGE

Assembly language is the fastest and most powerful language for computer and peripheral device.

Herewith we attached a AD14 program written by Assembly language so that the reader can use the ADDA-14 card more fluently under the program.

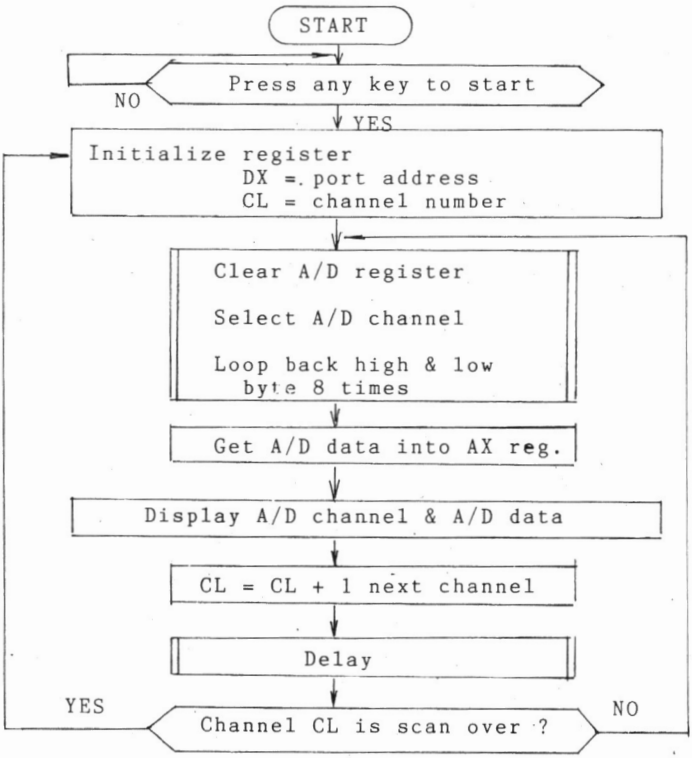
The assembly language was compile with IBM PC Macro-assembler. The speed is serval times faster then BASIC LANGUAGE.

By the way, please type ADINSTALL before using the AD14 assembly language.

EG.

*ADINSTALL ↵

9. Flow chart of AD14 (ASSEMBLY LANGUAGE)



```
 1:*; IBM-PC  ADDA-14  A/D Convert
 2: ;        Assembly Language
 3: ;
 4: ; ************************************************
 5: ; Define string character
 6: ;
 7: DATASEG SEGMENT ; data segment
 8: ADTIL_1 DB '  Channel =  $'
 9: ADTIL_2 DB '  Date =  $'
10: DATASEG ENDS     ; end of data segment
11: ;
12: ; ************************************************
13: PROGRAM SEGMENT ; code segment
14: ;
15: ; ------------------------------------------------
16: MAIN     PROC    FAR      ; main part of program
17:          ASSUME  CS: PROGRAM, DS: DATASEG
18: ;
19: START:                    ; starting execution address
20: ; Set up stack for return
21:          PUSH DS          ; save DS on stack
22:          SUB  AX,AX       ; set AX to 0
23:          PUSH AX          ; put AX to stack
24: KEYBOD:  MOV  AH,08H      ; check keyboard input funct

25:          INT  21H         ; call DOS
26:          CMP  AL,00H      ; is keyin ?
27:          JZ   KEYBOD      ; yes, keyin !
28:          CALL CRLF        ; skip a line
29: AD14:    MOV  DX,0171H    ; set DX port address
30:          MOV  CL,00H      ; CL = channel number
31; NXCHL:   CALL LPBACK      ; loop back 8 times
32:          PUSH DX          ; save DX
33:          MOV  DL,73H      ; get high byte address
34:          IN   AL,DX       ; get high 6 bit data
35:          AND  AL,3FH      ; mask high 2 bit
36:          MOV  AH,AL       ; move to high byte in AX
37:          DEC  DL          ; get low byte address
38:          IN   AL,DX       ; get low 8 bit data
39: ;
40: ; Now A/D convert data in AX register
41: ;
42:          PUSH AX          ; store AX
43:          PUSH CX          ; store CX
44:          MOV  AX,DATASEG ; set data to work area
45:          MOV  DS,AX       ;
46:          MOV  DX,OFFSET ADTIL_1
47:          MOV  AH,09H      ; display function
48:          INT  21H         ; call DOS
```

```
49:           POP   CX              ; pop CX
50:           MOV   BX,CX           ; channel number to BX
51:           CALL  BIN2DEC         ; call display BX routine
52:           PUSH  CX              ; store CX
53:           MOV   AX,DATASEG      ; set data to work area
54:           MOV   DS,AX           ;
55:           MOV   DX,OFFSET ADTIL_2
56:           MOV   AH,09H
57:           INT   21H             ; call DOS
58:           POP   CX              ; pop CX
59:           POP   AX              ; pop AX
60:           MOV   BX,AX           ; A/D data to BX
61:           CALL  BIN2DEC         ; call display BX routine
62:           CALL  CRLF            ; carriage return & linefeed
63:           CALL  DELAY           ; delay CX = FFFF
64:           POP   DX              ; pop DX
65:           INC   CL              ; next channel
66:           CMP   CL,10H          ; is 15 channel ?
67:           JNZ   NXCHL           ; yes !
68:           CALL  CRLF            ; carriage return & linefeed
69:           CALL  CRLF            ; carriage return & linefeed
70:           JMP   AD14            ; next routine
71:           RET                   ; return to DOS
72: MAIN      ENDP                  ; end of main procedure

73: ;
74: ; ---------------------------------------------------
75: ; Loop back 8 times subroutine
76: ;
77: LPBACK    PROC     NEAR         ; define procedure
78:
79:           PUSH DX               ; save DX
80:           PUSH CX               ; save CX
81:           MOV  AL,00H           ; clear AX
82:           OUT  DX,AL            ; clear A/D register
83:           DEC  DL               ; decrement DX
84:           MOV  AL,CL            ; channel number into AL
85:           OUT  DX,AL            ; select A/D channel
86:           MOV  DL,7CH           ; select low byte 8 bit
87:           MOV  CX,08H           ;  loop back 8 times
88: LOWBYTE:                       ;   for low byte
89:           IN   AL,DX            ;
90:           LOOP LOWBYTE          ;
91:           MOV  DL,78H           ; select high byte 6 bit
92:           MOV  CX,08H           ;  loop back 8 times
93: HIBYTE:                        ;   for high byte
94:           IN   AL,DX            ;
95:           LOOP HIBYTE           ;
96:           POP  CX               ; pop CX
```

```
 97:          POP  DX              ; pop DX
 98:          RET                  ; return from LPBACK
 99: LPBACK   ENDP                 ; end of procedure
100: ;
101: ; ---------------------------------------------------------
102: ; Convert binary in BX to decimal in screen
103: ;
104: BIN2DEC PROC     NEAR         ; define procedure
105:          PUSH CX              ; save CX
106:          MOV  CX,10000D       ; divide by 10000
107:          CALL DEC_DIV
108:          MOV  CX,1000D        ; divide by 1000
109:          CALL DEC_DIV
110:          MOV  CX,100D         ; divide by 100
111:          CALL DEC_DIV
112:          MOV  CX,10D          ; divide by 10
113:          CALL DEC_DIV
114:          MOV  CX,1D           ; divide by 1
115:          CALL DEC_DIV
116:          POP  CX              ; pop CX
117:          RET                  ; return form BIN2DEC
118: ;
119: ; ---------------------------------------------------------
120: ; Sub-subroutine to divide number in BX by
121: ;   number in CX, print in screen
122: ;
123: DEC_DIV PROC     NEAR         ; define procedure
124:          MOV  AX,BX           ; number high half
125:          MOV  DX,00H          ; zero out low half
126:          DIV  CX              ; divide by CX
127:          MOV  BX,DX           ; remainder into BX
128:          MOV  DL,AL           ; quotient into DL
129:          ADD  DL,30H          ; convert to ASCII
130:          MOV  AH,02H          ; display function
131:          INT  21H             ; call DOS
132:          RET                  ; return from DEC_DIV
133: DEC_DIV ENDP                 ; end of procedure
134: ;
135: ; ---------------------------------------------------------
136: ;
137: BIN2DEC ENDP                 ; end of procedure BIN2DEC
138: ;
139: ; ---------------------------------------------------------
140: ; Carriage return & Linefeed
141: ;
142: CRLF     PROC     NEAR
143:          PUSH CX              ; save CX
144:          MOV  DL,0DH          ; carriage return
```

```
145:            MOV   AH,02H       ; display function
146:            INT   21H          ; call DOS
147:            MOV   DL,0AH       ; linefeed
148:            MOV   AH,02H       ; display function
149:            INT   21H          ; call DOS
150:            POP   CX           ; pop CX
151:            RET                ; return from CRLF
152: CRLF       ENDP               ; end of procedure
153: ;
154: ; ---------------------------------------------------------------
155: ; Delay subroutine, delay time CX = FFFF
156: ;
157: DELAY      PROC     NEAR      ; define procedure
158:            PUSH  CX           ; save CX
159:            MOV   CX,9FFFH     ; define delay CX = FFFFH
160: NXDEL:     LOOP  NXDEL        ; is zero ?
161:            POP   CX           ; pop CX
162:            RET                ; return from delay
163: DELAY      ENDP               ; end of procedure
164: ;
165: ; ---------------------------------------------------------------
166: ;
167: PROGRAM ENDS                  ; end of code segment
168: ;

169: ; ***************************************************
170: ;
171:            END       START    ; end of main porgram
172: ;
```

10. THE THERMAL SENSOR

We are going to use PC ADDA-14 as a thermal sensor. The only thing we have to do is connect a TCV (Temperature Convert Voltage) circuit to our PC ADDA-14. Figure I shows the connection block diagram.

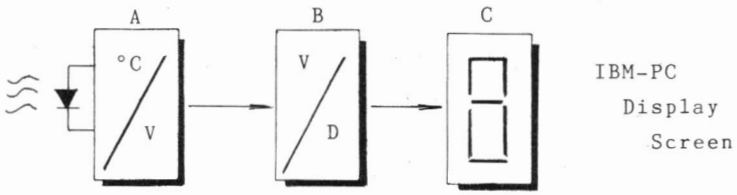


Figure I

As we can see in Fig. 3. This circuit works as a Temperature -voltage converter. There is a "Vref " printed at the left hand upper corner of Fig. 3. It is a reference voltage for the circuit. This voltage will affect the accurance of our sensor. It is advised to conncect a 7.15V reference voltage to this point.

Fig. 2 shows two methods to make a thermal sensor. We always use silican type semiconductor. It has better to use shielding cable as the enlongation wire. This kind of cable can stop extenal noise from jamming our circuit.

Fig. 4 Temperature - voltage conversion charateristic curve. From the curve we know that at 50°C, the output must at 8.5V.

11. PROCEDURE TO ADJUST THE WORKING RANGE

1. Immerse the sensor in 0°C water. use a multi-meter to read the voltage value of Pin 14 of LM324. Adjust the variable resistor P2 until the reading is 0.1 V.

2. As last, immerse the sensor in 50°C water. And then adjust P3 until the reading is 8.5V.

After you have completed the above steps, then the TCV circuit can direct connect the our PC ADDA-14.

Digitize Temperature Sensor is only one of the many usages of our PC ADDA-14. It all depends on your usage and immegination. Good luck !

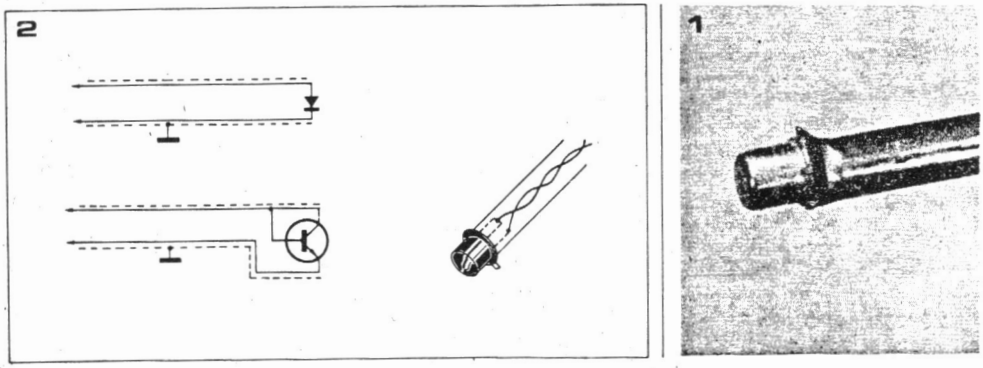


Fig. 1,2 transistor or diode thermal sensor

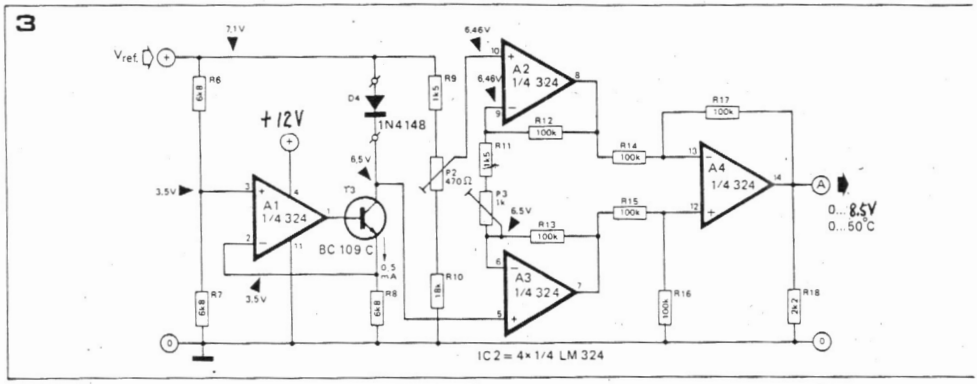


Fig.3 temperature - voltage convert circuit

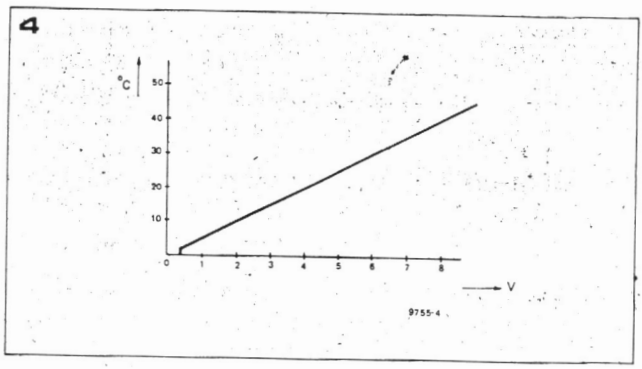


Fig. 4 "Temp - Volt " conv. char.